

AZ
Ifw

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

INVENTORS: Stephan L. JOURDAN, et al
SERIAL NO: 09/608,624
FILING DATE: June 30, 2000
TITLE: TRACE INDEXING VIA TRACE END ADDRESSES
ART UNIT: 2183
EXAMINER: Henry TSAI



Mail Stop Appeal Brief - Patents
COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, VA 22313-1450

SUPPLEMENTAL APPEAL BRIEF

SIR:

This supplemental brief is in response to the Notification of Non-Compliant Appeal Brief mailed October 30, 2006.

Remarks

The above-identified Notification alleges that the supplemental brief filed July 21, 2006 "does not contain arguments of the appellant with respect to each ground of rejection presented for review, and the basis therefor, with citations of the statutes, regulations, authorities, and parts of the record relied on as required by 37 CFR 41.37(c)(1)(vii). Appellant's remarks do not correspond to the current rejection of claim 1 under 102 with the Agarwal reference. Appellant's Appeal Brief discusses [sic] how I2 is not an entry point, but the rejection makes no mention of I2."

In response, a new supplemental appeal brief addressing all items required under § 41.37 is set forth below.

I. Real Party in Interest

The real party in interest is the Assignee, Intel Corporation.

II. Related Appeals and Interferences

There are no other appeals or interferences known to Appellant, Appellant's legal representative, or Assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in this Appeal.

III. Status of Claims

The application as filed included claims 1-19. Claims 20-27 were added in an amendment filed September 4, 2003. Claims 8 and 21 were canceled in an amendment filed April 8, 2004. Claims 31-37 were canceled and claims 38-43 were added in an amendment filed October 4, 2004. Claims 4-7, 9-15, 17-19, 23-27 and 41-43 were allowed, but their allowance was withdrawn in the Office Action mailed September 21, 2005. Claims 28-30 were canceled in the supplemental appeal brief filed July 21, 2006. Claims 15 and 39 are objected to. Claims 1-7, 9-20, 22-27 and 38-43 are rejected and herein on appeal. (Note: due to an apparent typographical error in the Office Action mailed April 5, 2006, which was carried over to the supplemental brief filed July 21, 2006, the pending claims were mistakenly identified as 1-7, 9-20, 23-27 and 38-43. They should have been identified as 1-7, 9-20, **22-27** and 38-43. Claim 22 was never canceled.)

IV. Status of Amendments

The amendments filed March 4, 2005 in response to the final rejection mailed January 4, 2005 have been entered. It is assumed that the amendment of the supplemental appeal brief filed July 21, 2006 (the cancellation of claims 28-30) has been entered, but no confirmation of this is available at the time of this writing.

V. Summary of Claimed Subject Matter

(References are to the specification and drawings.)

Independent claim 1.

Apparatus, comprising a memory entry storing a trace having a multiple-entry, single exit architecture.

Explanation.

Embodiments of the present invention relate to an “extended block” architecture for instructions in a processor. The extended blocks are types of “traces.” An extended block, according to embodiments of the present invention, may have multiple entry points but only a single exit point. (Page 2, lines 23-25.) Among other advantages, the traces provide for lessened redundancy, may be dynamically extended and may include shared blocks. (Page 3, lines 1-6.) The traces may be stored in an entry in a memory, such as a block cache 280. (Page 3, line 20 and FIG. 2.)

Examples of traces according to embodiments of the present invention are illustrated in FIGs. 4-6. As can be seen in these figures, the traces may have multiple entry points but only a single exit point. For example, FIG. 4(c) shows a trace with two entry points IP₃ and IP₄, with a single exit point IP₁.

Independent claim 4.

A front-end system for a processor, comprising:

an instruction cache system;

an extended block cache system, comprising:

a fill unit coupled to the instruction cache system,

a block cache,

a block predictor to store masks associated with complex blocks, the masks distinguishing block prefixes from each other; and

a selector coupled to the output of the instruction cache system and to an output of the block cache.

Explanation.

Embodiments of the invention include may include an instruction cache system. (Page 3, lines 9-10; FIG. 2, instruction cache 210.) The embodiments may further include an extended block cache system, comprising a fill unit coupled to the instruction cache system, a block cache, a block predictor and a selector. (Page 3, lines 7-29; FIG. 2, fill unit 260, block predictor 270, block cache 280 and selector 290.) The block cache may store extended blocks according to the invention, i.e., a type of trace having multiple entry points but only a single exit point. (Page 3, line 20; page 2, lines 23-25.) The block predictor may predict which extended blocks, if any, are likely to be executed and may cause the block cache to furnish any predicted blocks to an execution unit. (Page 3, lines 20-22.) The block predictor may store masks identifying whether a corresponding extended block is a complex or a non-complex block. A mask permits the block predictor to determine the position of an entry point to an extended block based on an instruction's instruction pointer. (Page 6, lines 24-30 to page 7, lines 1-2.) The selector may select which source, the instruction cache system or the block cache, supplies instructions to the execution unit. (Page 3, lines 26-28.)

Independent claim 9.

A method of managing extended blocks, comprising:

*predicting an address of a terminal instruction of an extended block to be used,
determining whether the predicted address matches an address of a terminal
instruction of a previously created extended block, and
selecting one of the extended block in the event of a match.*

Explanation.

Embodiments of the present invention may include a method of managing extended blocks (extended blocks have been previously described). According to the method, a block predictor may determine whether an address of a terminal instruction of an extended block can be predicted. (Page 4, lines 15-18; FIG. 2, block predictor 220; FIG. 3, flowchart blocks 1010, 1020.) Based on a "hit" of the address in a block cache, an extended block may be retrieved from the block cache and forwarded to an execution unit. (Page 4, lines 18-22; FIG. 2, block cache 280; FIG. 3, flowchart blocks 1030, 1040.)

Independent claim 16.

A processing engine, comprising:

*a front end stage storing blocks of instructions in a multiple-entry, single exit architecture when considered according to program flow, and
an execution unit in communication with the front end stage.*

Explanation.

Instruction blocks having a multiple-entry, single exit architecture have been discussed above. As further discussed above, the instruction blocks may be stored in a memory, which in turn may be part of a processor front end stage. In particular, the memory may be a block cache 280 that is part of an extended block cache ("XBC") 220 that communicates with an execution unit. (Page 3, lines 7-8 , line 20, line 26 and FIG. 2.)

Independent claim 17.

A processing engine, comprising:

*a front end stage storing blocks of instructions in a multiple-entry, single exit architecture when considered according to program flow, and
an execution unit in communication with the front end stage,
wherein the front-end stage includes
an instruction cache system,
an extended block cache system, including
a fill unit provided in communication with the instruction cache system,
a block cache, and
a selector coupled to the output of the instruction cache system and to an output of the block cache.*

Explanation.

Please see the explanation for claim 4.

Independent claim 20.

Apparatus, comprising a memory entry storing a sequence of program instructions as a trace, the instructions defining a program flow that progresses from any instruction therein to a last instruction in the memory entry and in which the trace has multiple separate prefixes.

Explanation.

A trace may be a sequence of instructions in program order. (Page 1, lines 17-19.) According to embodiments of the present invention, the sequence may have different prefixes but a common suffix, so that all of the prefixes lead to a common last instruction. (Page 6, lines 5-7 and FIG. 4(c).

Independent claim 23.

A memory comprising storage for a plurality of traces and means for indexing the traces by an address of a last instruction therein according to program flow.

Explanation.

Embodiments of the present invention may comprise a block cache for storing traces as explained earlier, for example, for claim 1. A trace may be indexed by an instruction pointer of a last instruction in the trace. (Page 2, line 25; page 4, lines 5-6.) An instruction pointer is a known technique for indexing instructions by address. (Page 3, lines 16-17.) Based on a "hit" of the instruction pointer in a block cache, a trace may be retrieved from the block cache and forwarded to an execution unit. (Page 4, lines 18-22.)

Independent claim 38.

Apparatus, comprising:

a memory having at least one memory entry; and

the at least one memory entry to store a trace having a multiple-entry, single exit architecture.

Explanation.

A trace may be a sequence of instructions in program order. (Page 1, lines 17-19.) According to embodiments of the present invention, the sequence may have a

multiple-entry, single exit architecture. (Page 2, lines 23-25.) The traces may be stored in a memory, such as a block cache 280. (Page 3, line 20 and FIG. 2.)

Independent claim 41.

Apparatus, comprising:

*a memory entry to store a trace in a single addressable memory location; and
means for indexing the trace by an address of a last instruction therein according
to program flow.*

Explanation.

Please see the explanation for claim 23.

Independent claim 42.

Apparatus, comprising:

*a memory having at least one memory entry;
the memory entry storing a trace in a single addressable memory location; and
means for indexing the trace by an address of a last instruction therein according
to program flow.*

Explanation.

Please see the explanation for claim 23.

Independent claim 43.

A memory, comprising:

*a trace stored in a single addressable memory location; and
means for indexing the trace by an address of a last instruction therein according to
program flow.*

Explanation.

Please see the explanation for claim 23.

VI. Grounds of Rejection to be Reviewed on Appeal

- A. Claims 1-3, 9-15, 20, 23-30 and 38-43 were rejected under 35 USC 101 as allegedly being directed to non-statutory subject matter. (Note: Claims 28-30 have been canceled.)
- B. Claims 23-37 and 41-43 were rejected under 35 USC 112, first paragraph, as allegedly failing to comply with the written description requirement. (Note: Claims 28-30 have been canceled.)
- C. Claims 1-3, 4-7, 20, 23-30 and 41-43 were rejected under 35 USC 112, second paragraph, as allegedly being indefinite. (Note: Claims 28-30 have been canceled.)
- D. Claims 1-3 and 38-40 were rejected under 35 USC 102(b) as allegedly being anticipated by Kaylor (US 5,492,276).
- E. Claims 1, 2, 16, 20, 28, 38 and 39 were rejected under 35 USC 102(e) as allegedly being anticipated by Agarwal (US 5,966,541). (Note: Claim 28 has been canceled.)

VII. Argument

- A. The rejection under 35 USC 101 should be withdrawn because the claims recite statutory subject matter.

Claims 1-3, 9-15, 20, 23-30 and 38-43 were rejected under 35 USC § 101 as being directed to non-statutory subject matter. Specifically, the Examiner contends that the language “a memory entry storing a trace” as recited in claim 1, for example, is “descriptive material per se and is non-statutory because it is not capable of, by itself, causing functional change in the computer.” Office Action mailed April 5, 2006 (hereafter, “Office Action”) at page 3, 3rd paragraph.

The § 101 rejection is error. First, removing any possible ambiguity, claim 1 recites an “[a]pparatus comprising a memory entry storing a trace” Thus, claim 1 recites at least a machine or an article of manufacture and is therefore statutory.

Further, the Examiner errs in stating that a memory entry storing a trace cannot cause a functional change in a computer. To the contrary, a trace is a sequence of computer instructions; changing the state of a computer is exactly what computer instructions do.

Independent claim 20 recites an “[a]pparatus, comprising a memory entry storing a sequence of program instructions as a trace” Thus, the preceding remarks are equally applicable to claim 20.

Independent claim 23 recites a “memory comprising storage for a plurality of traces” The person of ordinary skill in the computer-related arts understands a memory to be a component of a computer for electronically storing data. Many dictionaries recognize this meaning; one example is the popular online dictionary www.dictionary.com, which defines a memory as “a unit of a computer that preserves data for retrieval.” A trace, as noted earlier, is a sequence of computer instructions that change the state of a computer.

Independent claim 38 recites an “[a]pparatus, comprising: a memory ... and ... a trace” Similarly, independent claims 41-43 each recites an apparatus comprising a memory or memory entry storing a trace, or a memory storing a trace.

In view of the above, each of independent claims 1, 20, 23, 38 and 41-43 recites at least a machine or article of manufacture storing a sequence of instructions to change a state of a computer, and therefore recites statutory subject matter under § 101.

The Examiner rejects claim 9 as relating to “just an abstract idea.” The Examiner argues: “The claim does not provide practical application that produces a useful, tangible and concrete result. Therefore, this claim is non-statutory.” Office Action at page 4, 2nd paragraph.

This is error. As stated by the Federal Circuit,

every step-by-step process, be it electronic or chemical or mechanical, involves an algorithm in the broad sense of the term. Since § 101 expressly includes processes as a category of inventions which may be patented and § 100(b) further defines the word

"process" as meaning "process, art or method, and includes a new use of a known process, machine, manufacture, composition of matter, or material," it follows that it is no ground for holding a claim is directed to nonstatutory subject matter to say it includes or is directed to an algorithm. This is why the proscription against patenting has been limited to mathematical algorithms

In re Iwahashi, 888 F.2d 1370, 1374, 12 USPQ2d 1908, 1911 (Fed. Cir. 1989) (emphasis in the original). Claim 9 clearly does not set forth a mathematical algorithm. Instead, it sets forth a step-by-step process that does not fall within the "abstract idea" exception to patentable subject matter.

It is further observed that the Federal Circuit held in State Street Bank & Trust Company v. Signature Financial Group, 149 F.3d 1368, 47 USPQ2d 1596 (Fed. Cir. 1998) that "(...) the transformation of data, representing discrete dollar amounts, by a machine through a series of mathematical calculations into a final share price, constitutes a practical application of a mathematical algorithm, formula, or calculation, because it produces 'a useful, concrete and tangible result' -- a final share price momentarily fixed for recording and reporting purposes and even accepted and relied upon by regulatory authorities and in subsequent trades." In the present invention according to claim 9, a method is applied to accomplish the useful, concrete and tangible result of selecting an extended block which can then be forward to an execution unit.

In view of the above, the independent claims are statutory, as are the dependent claims for at least the reason that they incorporate the limitations of the independent claims.

B. The rejection under 35 USC 112, first paragraph, should be withdrawn because the claims clearly comply with the written description requirement.

Claims 23-27 and 41-43 were rejected under 35 USC 112, 1st paragraph as failing to comply with the written description requirement. Specifically, the Examiner alleges that claims 23 and 41-43 recite indexing means but that the specification contains no supporting description. Office Action, page 7, 2nd paragraph.

This is error. The specification has ample support for the rejected claims. For example, the specification in the paragraph bridging pages 2 and 3 reads, "Embodiments of the present invention assemble a new type of traces [sic], called

'extended blocks' herein, according to an architecture that permits several entry points but only a single exit point. These extended blocks *may be indexed based upon the address of the last instruction therein* " (emphasis added). The last sentence of the 3rd paragraph of page 3 reads, "Again, the block cache 280 may *index the extended blocks based upon an IP of the terminal instruction in the block*" (emphasis added). Claim 3 as originally filed reads, "The trace of claim 1, wherein the trace is *indexed by an address of a terminal instruction therein*" (emphasis added).

C. The rejection under 35 USC 112, second paragraph, should be withdrawn because the claims are definite.

Claims 1-3, 4-7, 20, 23-30 and 41-43 were rejected under 35 USC 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which the applicant regards as the invention. This is error.

Regarding claim 1 in particular, the Examiner contends as follows: "an 'entry' is ... nothing more than *a unit of information*. Thus, it is not clear how claim 1 constitutes an apparatus and how a memory entry can store anything" (emphasis in original).

Office Action, the paragraph bridging pages 7 and 8.

In response, it is observed that claim 1 recites not merely an "entry" but a "memory entry." As discussed previously, a memory is a unit of a computer that preserves data for retrieval. A memory entry, thus, is a sub-unit of a computer able to store at least a subset of the data that the memory as a whole can store.

Further concerning claim 1, the Examiner alleges that (1) "it is not clear how a 'trace' can have a multi-entry and a single exit" and that (2) "[t]he flow of a program can only have one entry in a trace." Office Action, page 8, 2nd paragraph. In response to (1), it is observed that the Examiner has applied a § 112, 2nd paragraph rejection here. It is not the role of the claims, under § 112, 2nd paragraph, to explain or describe the invention, but to "particularly point out and distinctly claim" the invention. Claim 1 does so.

The meaning of (2) is obscure. One possible interpretation is "the flow of a program can have entries, but only one of them can be in a trace." If this is what is

meant, the Appellant's response is that it does not appear to correspond in an understandable way with anything in the present specification or claims.

According to another possible interpretation, (2) would seem to subsume a program within a trace. In other words, another possible interpretation is "when a program is in a trace, the flow of the program can have only one entry point." The latter, again, does not appear relevant to the present specification or claims, which relate to a trace architecture with multiple entries and a single exit. If the Examiner is attempting to argue that no such trace architecture can exist, the Appellant's response is that the Examiner is mistaken: the present application discloses and claims just such a trace architecture.

In support of (2), the Examiner argues that (3) "[t]he flow of a program can only have one entry in a trace ... since the trace is dependent on the input to the program." Office Action, page 8, 2nd paragraph. As a basis for the foregoing, the Examiner cites "Nair Ravi, 'Trace Caches and Trace Processors', lecture note, Pages 1-2, USPTO training for examiners in the computer architecture area, 4/29/04" (hereafter, "Ravi"). Id.

The cited portion of Ravi discloses that a "trace of a program is the exact sequence of instructions executed by the computer while running the program," and that "[f]or a given program, the trace is dependent on the input to the program." Ravi, page 1. The latter is repeated, as noted above, by the Examiner in (3).

The Appellant responds that in (3) the Examiner juxtaposes two entirely unrelated ideas. Ravi's "the trace is dependent on the input to the program" means nothing more than what it plainly says: that a trace depends on a program's input. This places no restrictions whatsoever on, and in fact has nothing at all to do with, how a trace is architected in terms of entries and exits.

Concerning claim 3, the Examiner states,

"In claim 3, it is not clear what is meant by 'terminal instruction therein'. It appears that the terminal instruction is inside a trace stored in the memory entry. However, a trace comprises the sequences of executed instructions (see page 1, lines 16 in the specification), e.g., it comprises the address data (or pointers) of executed instructions. The address data is not an instruction. Therefore, a trace does not have terminal instruction therein."

Office Action, paragraph bridging pages 8 and 9. To point out only the most obvious error and self-contradiction in this paragraph: the Examiner observes that a trace comprises a sequence of executed instructions, and then concludes that it does not have a terminal instruction. This is impossible.

Concerning claim 4, the Examiner states, "it is not clear how to define 'complex blocks' and 'block prefixes' since the structural relationship between 'complex blocks', 'block prefixes', and the front-end system was not defined." Office Action, page 9, 2nd paragraph. Here, again the Examiner seeks to force the role of the written description onto the claims, apparently taking the position that terms whose meanings are self-evident need to be bolstered with added description. As discussed previously, this is not required of the claims. Notwithstanding -- a "block prefix" is simply that -- a prefix of a block. The prefixes are for blocks labeled "complex." The relationship between the complex blocks, their prefixes and the front-end system is that the complex blocks and block prefixes are elements of the front-end system, which is perfectly clearly from the fact that the language reciting the complex blocks and block prefixes follows "A front-end system for a processor, comprising:".

As to claim 6, the Examiner alleges that "it is not clear what is meant by 'blocks having a multiple-entry, single exit architecture'. Some essential elements or more detailed descriptions are missing." Office Action, page 9, 3rd paragraph. The Appellant again points out that "detailed descriptions" are not a statutory requirement for the claims.

As to claim 20, the Examiner states that "it is not clear what is meant by 'a last instruction in the memory entry'," and repeats the self-contradictory argument made concerning claim 3, i.e., that a sequence of instructions does not have a last instruction. See Office Action, paragraph bridging pages 9 and 10. The Appellant reiterates that this is error.

Further concerning claim 20, the Examiner states that "it is not clear how a trace has multiple separate prefixes." Office Action, page 10, 2nd paragraph. The Appellant repeats that the claims are not required to explain the invention. In support of the proposition that "it is not clear how a trace has multiple separate prefixes," the Examiner repeats the contention, made earlier in connection with claim 1, that "[t]he

flow of a program can only have one entry in a trace," and again cites the Ravi reference. Id. As these issues were addressed by the Appellant earlier in the discussion of the § 112, 2nd paragraph rejection of claim 1, the Appellant refers the reader to these earlier remarks.

Regarding claim 23, the Examiner states that "it is not clear how to define 'a last instruction therein' since there may have [sic] many last instructions in the memory." Office Action, page 10, 3rd paragraph. In claim 23, "a last instruction therein" refers to a last instruction of "traces," not of a memory.

Regarding claim 27, the Examiner states that "it is not clear how a trace can include executable instructions. As set forth above, a trace comprises the sequence of executed instructions." Office Action, paragraph bridging pages 10 and 11. As others of the Examiner's statements have been, the latter statements are self-contradictory: in the same breath, the Examiner says that a trace doesn't include instructions, and then that it does. This is error.

Concerning claims 23 and 41-43, the Examiner states that these claims "recited indexing means, however, in the specification there's no description about the structure for the indexing means necessary for supporting the claims." Office Action, page 11, 2nd paragraph. In response, the Appellant observes that the Examiner has applied a § 112, 2nd paragraph rejection here. The Examiner's comments, however, appear to relate to the requirements of § 112, 1st paragraph. In any event, ample support for indexing means has been previously demonstrated earlier.

D. The rejection of claims 1-3 and 38-40 under 35 USC 102(b) as allegedly being anticipated by Kaylor (US 5,492,276) should be withdrawn because Kaylor does not disclose any of the claim limitations.

To anticipate a claim under § 102, a single prior art reference must identically disclose each and every claim element. See Lindeman Maschinenfabrik v. American Hoist and Derrick, 730 F.2d 1452, 1458 (Fed. Cir. 1984). If any claimed element is absent from a prior art reference, it cannot anticipate the claim. See Rowe v. Dror, 112 F.3d 473, 478 (Fed. Cir. 1997).

Claims 1-3 and 38-40 were rejected under 35 USC § 102(b) as being anticipated by Kaylor (U.S. 5,492,276) ("Kaylor"). This rejection constitutes unmitigated error. Not only does the Kaylor reference not disclose the elements of the rejected claims as required under § 102, it is not even remotely related to the same field as the present invention. The present invention, as disclosed and claimed, relates to computers, and in particular to processor front-ends and techniques for managing them. The Kaylor reference, by contrast, relates to plumbing. Not a single element in the Kaylor reference can fairly be found under any reasonable interpretation to correspond to any element of the rejected claims.

E. The rejection of claims 1, 2, 16, 20, 28, 38 and 39 under 35 USC 102(e) as allegedly being anticipated by Agarwal (US 5,966,541) should be withdrawn because Agarwal does not disclose any of the claim limitations.

Independent claim 1.

Discussion begins with independent claim 1. Agarwal does not anticipate claim 1 for at least the reason that Agarwal does not disclose a memory **entry** that stores a trace having a multiple-entry, single exit architecture as recited in claim 1.

Agarwal contains no disclosure of a relationship between a memory entry and a trace. Instead, Agarwal's disclosure relates to software generally, and more particularly to an arrangement for "patching" faulty software, especially for the Year 2000 problem. In view of this, as an equivalent to the claimed memory entry, the Examiner can offer only "the memory space containing blocks 101, 102 and 103 as shown in Fig. 8" (Office Action, page 13, item 12, lines 6-7). But this is hardly adequate; claim 1 requires that the trace be stored in a **memory entry**, not merely memory generally. Agarwal does not disclose this subject matter. Agarwal describes no relationship between his blocks and memory, whether they fit in single entries or whether they are distributed across multiple memories. This isn't surprising since Agarwal is devoted to finding solutions to the Year 2000 problem – any relationship between his blocks 101-103 and memory entries is immaterial to the performance of his processes.

Further, it is observed that Agarwal does not mention traces at all. A trace is a specific code structure that occurs in processors. This structure cannot be found in Agarwal. Notwithstanding, the Examiner alleges that blocks 101-103 shown in FIG. 8 of Agarwal are equivalent to the claimed trace. See the Office Action, page 13, item 12, lines 10-11: "... storing a trace (an extended block including blocks 101, 102 and 103 as shown in Fig. 8 ...)" (emphasis in original). However, even assuming solely for purposes of argument that Agarwal's blocks 101, 102 and 103 correspond to the claimed trace, the blocks 101-103 at best have a *single-entry, single-exit* architecture. This is evident from FIG. 8 of Agarwal, an annotated version of which is shown below.

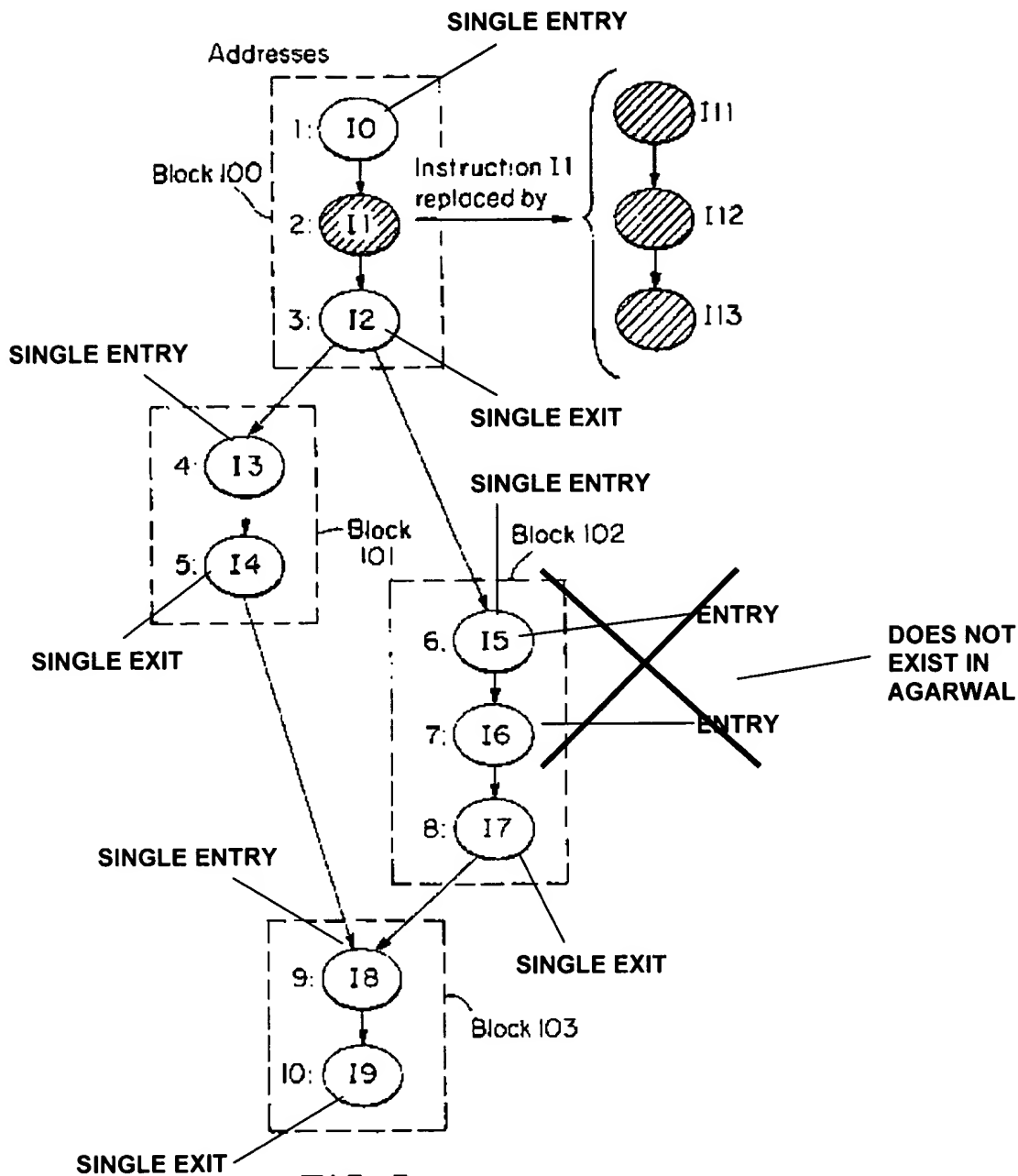


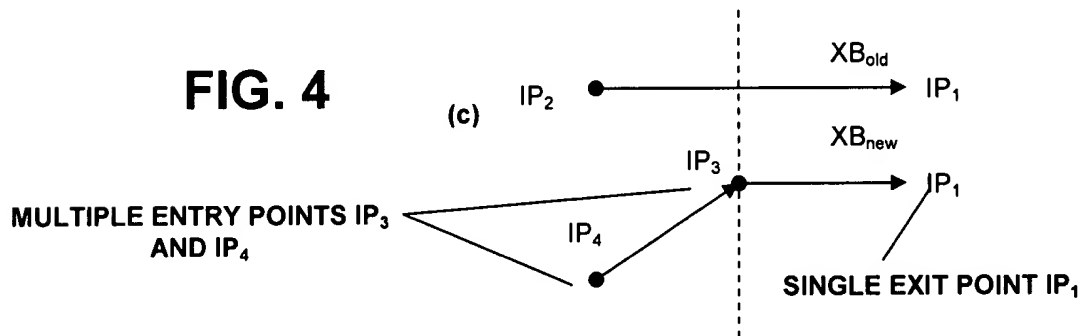
FIG. 8
(Agarwal, US 5,966,541)

In the annotated FIG. 8, it can be seen that Agarwal dissects the code into separate blocks 100, 101, 102 and 103, each of which have a single entry, single exit architecture. Specifically, block 100 has a single entry point at I0 and a single exit point

at I2, block 101 has a single entry point at I3 and a single exit point at I4, and so on. Nowhere do the blocks exhibit a multiple entry structure, as illustrated hypothetically on block 102. Agarwal offers no teaching or suggestion to consider the code any other way.

In light of the above, it is apparent that the Examiner's strained effort to find multiple entry points in Agarwal's blocks 100-103 falls short. The Examiner refers specifically to I3 and I5 as multiple entry points (e.g., in the Office Action, page 13, item 12, line 13). However, under Agarwal's scheme, I3 and I5 are the single entry points, respectively, of blocks 101 and 102.

An annotated FIG. 4(c) of the present application, below, shows an example of the claimed multiple-entry, single-exit architecture by way of contrast to Agarwal's blocks 100-103. In FIG. 4(c), a trace has multiple entry points IP₃ and IP₄, and a single exit point IP₁.



In view of the foregoing, claim 1 is allowable over Agarwal.

Dependent claim 2.

Rejected claim 2 depends on claim 1 and is therefore likewise allowable over Agarwal for at least the reasons discussed in connection with claim 1.

Independent claim 16.

Agarwal does not anticipate claim 16 for at least the reason that Agarwal does not disclose a front end stage storing blocks of instructions in a multiple-entry, single exit architecture when considered according to program flow. Moreover, Agarwal is completely silent as to a front end stage storing such blocks, and as to an execution unit in communication with the front end stage.

Agarwal's deficiencies concerning the multiple-entry, single exit architecture of embodiments of the present invention have been discussed above. As to the claimed front end stage in communication with an execution unit, Agarwal discloses only software structures with any specificity. Disclosure of hardware goes no deeper than "some memory" (col. 11, lines 36-38). Accordingly, Agarwal cannot meet the recitations of claim 16. Claim 16 is therefore allowable over Agarwal.

Independent claim 20.

Agarwal does not anticipate claim 20 for at least the reason that Agarwal does not disclose a trace with multiple separate prefixes as recited. This feature is analogous to the multiple entry feature discussed above, and which has been amply demonstrated to be absent from Agarwal. Claim 20 is therefore allowable over Agarwal.

Dependent claim 22.

As noted above, claim 22 is pending and is allowable for at least the reason that it depends on allowable claim 20.

Independent claim 38.

Finally, Agarwal does not anticipate claim 38 for at least the reason that Agarwal does not disclose a trace having a multiple-entry, single exit architecture as discussed above. Claim 38 is therefore allowable over Agarwal..

Dependent claim 39.

Claim 39 is allowable for at least the reason that it depends on allowable claim 38.

Claim objections

Claim 15 was objected to as being informal. This is error. Claim 15 refers back to the "selected block" of claim 9. There can be no confusion about which block is meant, since only one block is associated with a selecting step in claim 9.

Claim 39 was objected to as being of improper dependent form for failing to further limit the subject matter of a previous claim. This is error. Claim 38 recites "a trace having a multiple-entry, single exit architecture." Claim 39 further limits the subject matter of claim 38 by specifying that the trace is a "complex" trace and that it has "multiple independent prefixes and common, shared suffix." This information imparts further specificity and structure to the trace recited in claim 38.

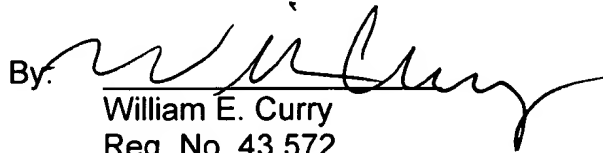
Conclusion

In view of the above, it is clear that the Examiner erred in rejecting claims 1-7, 9-20, 22-27 and 38-43, and in objecting to claims 15 and 39. It is therefore respectfully requested that the Board reverse the Examiner, withdraw all rejections and objections, and allow claims 1-7, 9-20, 22-27 and 38-43.

The Examiner is invited to contact the undersigned at (202) 220-4323 to discuss any matter concerning this application. The Office is authorized to charge any fees related to this communication to Deposit Account No. 11-0600.

Respectfully submitted,

Dated: Nov. 30, 2006

By: 
William E. Curry
Reg. No. 43,572

KENYON & KENYON LLP
Attorneys for Intel Corporation
1500 K Street, N.W., Suite 700
Washington, D.C. 20005
Tel: (202) 220-4200
Fax: (202) 220-4201

APPENDIX
Claims on Appeal

1. Apparatus, comprising a memory entry storing a trace having a multiple-entry, single exit architecture.
2. The apparatus of claim 1, wherein the trace is a complex trace having multiple independent prefixes and a common, shared suffix.
3. The apparatus of claim 1, wherein the entry is indexed by an address of a terminal instruction therein.
4. A front-end system for a processor, comprising:
 - an instruction cache system;
 - an extended block cache system, comprising:
 - a fill unit coupled to the instruction cache system,
 - a block cache,
 - a block predictor to store masks associated with complex blocks, the masks distinguishing block prefixes from each other; and
 - a selector coupled to the output of the instruction cache system and to an output of the block cache.
5. The front-end system of claim 4, wherein the extended block cache system further comprises a block predictor coupled to the fill unit and the block cache.
6. The front-end system of claim 4, wherein the block cache is to store blocks having a multiple-entry, single exit architecture.
7. The front-end system of claim 4, wherein the block cache is to store complex blocks having multiple independent prefixes and a common suffix.
9. A method of managing extended blocks, comprising:
 - predicting an address of a terminal instruction of an extended block to be used,
 - determining whether the predicted address matches an address of a terminal instruction of a previously created extended block, and

selecting one of the extended block in the event of a match.

10. The method of claim 9, further comprising creating a new extended block when there is no match.

11. The method of claim 10, wherein the creating comprises:
receiving new instructions until a terminal condition occurs,
assembling the new instructions into an extended block,
determining whether an address of a terminal instruction in the new block matches an address of a terminal instruction of a pre-existing block, and
unless a match occurs, storing the new block in a memory.

12. The method of claim 11, wherein the storing comprises, when an older block causes a match, storing the new block over the old block in a memory if the old block is subsumed within the new block.

13. The method of claim 11, wherein the storing comprises, when an older block causes a match, dropping the new block if the new block is subsumed within the older block.

14. The method of claim 11, wherein the storing comprises, when an older block causes a match, creating a complex block if the new block and the older block share a common suffix but include different prefixes.

15. The method of claim 9, further comprising outputting instructions of the selected block for execution.

16. A processing engine, comprising:
a front end stage storing blocks of instructions in a multiple-entry, single exit architecture when considered according to program flow, and
an execution unit in communication with the front end stage.

17. A processing engine, comprising:
a front end stage storing blocks of instructions in a multiple-entry, single exit architecture when considered according to program flow, and

an execution unit in communication with the front end stage,
wherein the front-end stage includes
an instruction cache system,
an extended block cache system, including
 a fill unit provided in communication with the instruction cache system,
 a block cache, and
 a selector coupled to the output of the instruction cache system and to an output
of the block cache.

18. The processing engine of claim 17, wherein the block cache is to store the multiple-entry, single exit traces.

19. The processing engine of claim 17, wherein the extended block cache system further comprises a block predictor coupled to the fill unit and the block cache.

20. Apparatus, comprising a memory entry storing a sequence of program instructions as a trace, the instructions defining a program flow that progresses from any instruction therein to a last instruction in the memory entry and in which the trace has multiple separate prefixes.

22. The apparatus of claim 20, wherein the memory entry is indexed by an address of a terminal instruction therein.

23. A memory comprising storage for a plurality of traces and means for indexing the traces by an address of a last instruction therein according to program flow.

24. The memory of claim 23, wherein the traces include a plurality of instructions assembled according to program flow.

25. The memory of claim 24, wherein the traces have a multiple entry, single exit architecture.

26. The memory of claim 24, wherein at least one trace has separate prefixes and a common suffix, when considered according to program flow.

27. The memory of claim 24, wherein at least one trace includes at least three segments of executable instructions in which, when considered according to program flow, first and second segments are mutually exclusive of each other and lead into the third segment.

38. Apparatus, comprising:
a memory having at least one memory entry; and
the at least one memory entry to store a trace having a multiple-entry, single exit architecture.

39. The apparatus of claim 38, wherein the trace is a complex trace having multiple independent prefixes and a common, shared suffix.

40. The apparatus of claim 38, wherein the entry is indexed by an address of a terminal instruction therein.

41. Apparatus, comprising:
a memory entry to store a trace in a single addressable memory location; and
means for indexing the trace by an address of a last instruction therein according to program flow.

42. Apparatus, comprising:
a memory having at least one memory entry;
the memory entry storing a trace in a single addressable memory location; and
means for indexing the trace by an address of a last instruction therein according to program flow.

43. A memory, comprising:
a trace stored in a single addressable memory location; and
means for indexing the trace by an address of a last instruction therein according to program flow.

EVIDENCE APPENDIX

No evidence has been submitted.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.